Santa Clara University
School of Engineering

CSEN 79-2: Mid-Term #2, Coding

Assignments

Finish the "BigNum" class, as provided in bignum.tar, with the "rule of 4" member functions (destructor, copy constructor, assignment operator, move operator), another constructor that accept a "`long`" argument, and the "+" operator. Design an automated test to cover the code execution paths and the boundary conditions. Submit the source code, including the test scripts, if you chose to write them.

The automated tests should exercise the code sufficiently and verify the results.

Background

An integral value can be mathematically represented by a polynomial.

$v = sign * \sum_{i=0}^{n} x_i B^i$, where sign = ± 1, $0 \leq x_i < B$ and B is the "base" which is an integer usually greater than or equal to 2.

For computer software, n is the length of the integral type that is 7, 15, 31, or 63 for types `char`, `short`, `int`, or `long`. For these types, the most significant bit represent sign where zero is positive and 1 negative.

This homework asks you to complete the design for a class, called **`BigNum`**, that effectively removes the length limitation for C++ integral types. The approach is a container with an internal storage that is extensible and utilize the polynomial mathematics for arithmetic operations.

Hints

The provided code can convert a string of decimal digits, effectively unlimited in length, to a BigNum object. It can also output the internal storage in the format suitable for verification. You should understand that algorithm and see if you can utilize it for the addition operator. To add two polynomials, one only need to loop through them and add the coefficients of the same order. Since we have a slot for each order, the code is relatively straight-forward. The tricky part is when the sum of the two coefficients exceeds the base. In that case, one must "carry" to the higher order and reduce the current one appropriately.

Operator+ can be a friend or a member function. The choice is personal. Pay attention to the function prototype syntax, as C++ compiler expect exact match.

Provided code's implementation of "rule of 4" members is incomplete.

Unpack bignum.tar into a directory then build it. It should yield an executable "bignum". Test it with the provided test data and observe the outcome. The program outputs Python compatible lines to the screen to be checked by your favorite Python interpreter. **Do not proceed without doing this step.**

Study the constructor that accepts a string and understand how "digits" materializes. Pay attention to "checkCapacity". It may throw exception and therefore making BigNum constructor to fail.

Santa Clara University
School of Engineering

CSEN 79-2: Mid-Term #2, Coding

Consider changing the storage of digits from pointer to vector. This should make the assignment easier to debug. That, however, requires you to also change the constructor with string as well as “checkCapacity.”

The code takes the incoming decimal string from left to right. After consuming the optional sign character, each digit effectively multiplies the integral value by 10, before adding the new character’s decimal value.

As digits get a new value, it is important to avoid overflow. We do so by using a buffer that’s twice as long as the digit. If we found the resulting value greater than the maximum of the storage unit, we “carry” the excess to the next higher order digit. This “compute and carry if necessary” operation is repeated or cascading up until we have done all the processing. The addition operation must do similarly.

The formatted output operator (<<) was crudely done. It outputs the digits in polynomial form really meant for debugging only. Consider implementing one of the prettier output in decimal, hexadecimal, or octal notations.